Общество с ограниченной ответственностью
«ОСК»
115114, г. Москва, ул. Ленинская слобода, д. 19
тел.: +7 (495) 609-60-88
www.mfms.ru

Сервис доставки push-сообщений

Push-платформа

Руководство по интеграции push-библиотеки в приложение на базе ОС Android

Москва 2017

ОГЛАВЛЕНИЕ

ТЕРМИНЫ И СОКРАЩЕНИЯ

Термин	Полная форма	Описание
appPackage		Уникальный код приложения заказчика в магазинах приложений Google Play и App Store. Используется при регистрации приложения на push-сервере
clientId		Идентификационный номер клиента в системах заказчика. Является дополнительным идентификатором приложения на push-сервере
confirmationCode		Код подтверждения регистрации приложения на push-сервере
deviceUid		Уникальный идентификатор мобильного приложения, установленного на конкретном устройстве. Формируется по заданному алгоритму при каждом запуске приложения на стороне приложения
deviceAddress		Уникальный адрес мобильного приложения, установленного на конкретном устройстве. Формируется по заданному алгоритму при каждом запуске приложения на стороне push-сервера. Может меняться при изменении push-адреса приложения в PNS. Это зависит от настроек, заданных для приложения на push-сервере
phoneNumber		Номер мобильного телефона клиента в формате E.164. Является дополнительным идентификатором приложения на push-сервере
PNS	Push Notification Services	Провайдеры push-уведомлений APNS (Apple Push Notification Service) и GCM (Google Cloud Message), обеспечивают доставку push-уведомлений в приложение на устройстве
Push-адрес		Уникальный адрес приложения в PNS, может изменять значение по инициативе PNS
Push-сервер		Программно-аппаратный комплекс, который реализует сервис доставки push-сообщений. В частности, механизм регистрации приложений на push-сервере, отправку push-уведомлений и доставку содержимого push-сообщений в приложение на устройстве, резервирование доставки с помощью sms-сообщений, хранение статусов доставки push-сообщений
Push-сообщение		Сообщение от заказчика, включающее push-уведомление и содержимое сообщения в текстовом формате. В качестве содержимого заказчик может передавать: тексты, а также бинарные данные в формате Base64
Push-уведомление		Короткое уведомление, которое push-сервер отправляет в приложение посредством сервисов PNS. Push-уведомления, доставленное в приложение, инициирует процедуру получения содержимого push-сообщения с push-сервера

Термин	Полная форма	Описание
Библиотека	Push-библиотека	Компонент мобильного приложения, участвует в интеграции с приложением. Обеспечивает регистрацию приложения на push-сервере, привязку дополнительных идентификаторов, доставку push-уведомлений и содержимого push-сообщений в приложение
Заказчик		Издатель мобильного приложения, является инициатором отправки push-сообщений
ИС		Информационные системы заказчика
Клиент		Владелец мобильного устройства с установленным мобильным приложением заказчика. Является конечным получателем push-сообщений
Приложение	Мобильное приложение	Программное обеспечение заказчика, установленное на мобильном устройстве клиента, в которое интегрирована push-библиотека

1. ВВЕДЕНИЕ

Данный документ представляет собой руководство разработчика, описывающее порядок работы по интеграции push-библиотеки в приложение заказчика. В документе приводится необходимая информация по интеграции, настройке и администрированию библиотеки.

Рабочий язык документа – русский, использование английских терминов и сокращений допускается.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Назначение

Библиотека предназначена для решения следующих задач в рамках сервиса доставки push-сообщений:

- регистрация приложения на push-сервере;
- привязка дополнительных идентификаторов к зарегистрированным приложениям;
- доставка push-уведомлений и push-сообщений в приложение;
- отправка файлов и сообщений из приложения;
- отслеживание статусов сообщений.

2.2. Область применения

Процедура интеграции библиотеки в приложение заказчика является частью процесса внедрения сервиса доставки push-сообщений.

3. ТРЕБОВАНИЯ К ИНТЕГРАЦИИ

Библиотека поддерживает приложения на базе ОС Android версии 4.0 и выше, а также требует подключения в приложении Google Play Services версии 10.0 и выше. В Google Play Service возможно подключить только модуль GCM или Google Play Service целиком (подробное описание подключения доступно по ссылке: https://developers.google.com/android/guides/setup).

4. ИНТЕГРАЦИЯ БИБЛИОТЕКИ В ПРИЛОЖЕНИЕ

4.1. Порядок настройки интеграции

Настройка интеграции библиотеки в приложение производится в следующей последовательности:

1. Обновить файл AndroidManifest.xml.
2. Добавить зависимости в файл build.gradle.
3. Реализовать классы PushBroadcastReceiver и PushServerIntentService.
4. Выполнить регистрацию приложения по deviceUid.
5. Выполнить привязку идентификаторов clientId и phoneNumber к зарегистрированному приложению.

4.1.1. AndroidManifest.xml

Для настройки интеграции библиотеки в приложение необходимо обновить файл «AndroidManifest.xml», добавив в него указанный программный код:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="[YOUR_APPLICATION_PACKAGE_NAME]">
   <permission
      android:name="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE"
      android:protectionLevel="signature"/>
   <uses-permission
android:name="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE"/>
   <application>
   <meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
   <!-- Setting up Push FCM service -->
   <meta-data
      android:name="com.pushserver.android.senderId"
      android:value="!STRING![FCM_SENDER_ID]" />
    <meta-data
      android:name="com.pushserver.android.service"
android:value="[YOUR_APPLICATION_PACKAGE_NAME].[IMPLEMENTATION_INTENT_SERVICE]" />
   <meta-data
      android:name="com.pushserver.android.providerUid"
      android:value="[YOUR_PROVIDER_UID]"/>
   <meta-data
      android:name="com.pushserver.android.logs"
      android:value="[LOGGING ENABLED]" />
   <meta-data
      android:name="com.pushserver.android.heartbeatInterval"
      android:value="[HEARTBEAT_INTERVAL]" />
   <meta-data
      android:name="com.pushserver.android.serverIdList"
      android:resource="[SERVER_ID_ARRAY]"/>
   <meta-data
      android:name="com.pushserver.android.serverUrlList"
      android:resource="[SERVER_URL_ARRAY]"/>
   <meta-data
      android:name="com.pushserver.android.serverPrimaryId"
      android:value="[PRIMARY_SERVER_ID]"/>
   <meta-data
      android:name="com.pushserver.android.useDefaultDeviceUID"
      android:value="[USE_DEFAULT_UID_GENERATOR]" />
   <meta-data
      android:name="com.pushserver.android.uploadServerUrl"
```

```
    android:value="[UPLOAD_SERVER_URL]"/>
    <receiver
       android:name="[YOUR_BROADCAST_RECEIVER]"
       android:exported="false"

    android:permission="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE
">
       <intent-filter>
          <action android:name="com.pushserver.android.NEW_PUSH_MESSAGE_EVENT" />
          <action android:name="com.pushserver.android.RECEIVER_CHANGED_EVENT" />
          <action  android:name="com.pushserver.android.DEVICE_ADDRESS_CHANGED_EVENT"
/>
          <action android:name="com.pushserver.android.HAS_SECURED_MESSAGE" />
          <action android:name="com.pushserver.android.ERROR_EVENT" />
       </intent-filter>
    </receiver>
    <service
android:name="[YOUR_APPLICATION_PACKAGE_NAME].[IMPLEMENTATION_INTENT_SERVICE]" />
    </application>
</manifest>
```

Используемые параметры:

– [FCM_SENDER_ID] – идентификатор FCM для регистрации приложения на push-сервере и получения push-уведомлений;

Примечание! Префикс «!STRING!» необходим всегда.

– [YOUR_BROADCAST_RECEIVER] – путь к классу-наследнику PushBroadcastReceiver, который будет обрабатывать события библиотеки;
– [YOUR_APPLICATION_PACKAGE_NAME] – полное наименование пакета приложения;
– [IMPLEMENTATION_INTENT_SERVICE] – имя класса, реализующего ServerIntentService;
– [YOUR_PROVIDER_UID] – идентификатор заказчика на push-сервере;
– [LOGGING ENABLED] – позволяет включать/отключать логирование библиотеки;
– [HEARTBEAT_INTERVAL] – интервал опроса GCM (в миллисекундах), рекомендованное значение – 300000 (5 минут);
– [SERVER_ID_ARRAY] – массив идентификаторов push-серверов, используемых для отправки push-сообщений;
– [SERVER_URL_ARRAY] – массив URL push-серверов, используемых для отправки push-сообщений;
– [PRIMARY_SERVER_ID] – значение идентификатора push-сервера по умолчанию;
– [USE_DEFAULT_UID_ GENERATOR] – логическая переменная, значение которй определяет по умолчанию способ генерации deviceUid:

- true – значение deviceUid автоматически генерируется библиотекой;
- false – значение deviceUid определяется заказчиком;

– [UPLOAD_SERVER_URL] – url-адрес сервера, на который отправляются сообщения и файлы из приложения.

4.1.2. Gradle

Необходимо обновить build.gradle на верхнем уровне и файлы build.gradle на уровне приложения, добавив следущие зависимости:

1. classpath 'com.google.gms:google-services:3.0.0':

```
buildscript {
   repositories {
      jcenter()
   }
   dependencies {
      classpath 'com.android.tools.build:gradle:2.3.0'
      classpath 'com.google.gms:google-services:3.0.0'
   }
}

allprojects {
   repositories {
      jcenter()
   }
}
```

2. compile project(path: ':push-api'):

```
apply plugin: 'com.android.application'
repositories {
   mavenCentral()
   flatDir {
      dirs 'libs'
   }
}
android {
   ...
}
dependencies {
   compile 'com.google.firebase:firebase-messaging:10.2.0'
   compile project(path: ':push-api')
}
apply plugin: 'com.google.gms.google-services'
```

4.1.3. Классы для реализации

Для дальнейшей настройки интеграции библиотеки в приложение требуется реализовать следующие классы:

– PushBroadcastReceiver – обработчик событий библиотеки (подробное описание методов класса представлено в разделе 4.1.3.1);

– PushServerIntentService – обработчик сохранения сообщений (подробное описание методов класса представлено в разделе 4.1.3.2).

4.1.3.1. Обработчик событий библиотеки PushBroadcastReceiver

Для обработки событий библиотеки в наследнике класса PushBroadcastReceiver требуется реализовать методы, представленные в Таблица 1.

Таблица 1. Методы класса PushBroadcastReceiver.

№ п/п	Метод	Описание	Условия вызова
1.	protected abstract void onNewPushNotification (android.content.Context context java.lang.String alert android.os.Bundle bundle)	Получение нового push-сообщения от FCM. Параметры: - context – контекст; - alert – текст PushNotification; - bundle – содержимое push-сообщения	Метод вызывается при получении нового PushNotification от FCM. В качестве параметра передается текст, полученный из shortMessage при отправке PNS (также Bundle с содержимым push-сообщения). Возможно отобразить в области уведомлений ОС
2.	protected abstract void onStatusChanged (android.content.Context context, java.lang.String registrationId)	Изменение статуса регистрации на FCM. Параметры: - context – контекст; - registrationId – идентификатор приложения в push-сети	Метод вызывается при изменении push-адреса в FCM
3.	protected abstract void onDeviceAddressChanged (android.content.Context context, java.lang.String newDeviceAddress)	Изменение значения deviceAddress Параметры: - context – контекст; - newDeviceAddress – новое значение deviceAddress	Метод вызывается после изменения значения deviceAddress

№ п/п	Метод	Описание	Условия вызова
4.	protected abstract void onDeviceAddressProblems (android.content.Context context, java.lang.String errorCode)	Не удалось изменит значение deviceAddress. Параметры: - context – контекст; - errorCode – код ошибки	Метод вызывается при наличии проблем, препятствующих генерации deviceAddress, и получает код проблемы: NO_GCM_ID – в случае отсутствия registrationId Google: - NO_CONTEXT – неправильная инициализация библиотеки, не передан контекст приложения; - NO_DEVICE_ID – не установлен идентификатор приложения; - BAD_ENCODING – отсутствует поддержка кодировки UTF-8 в приложении.
5.	protected abstract void onError (android.content.Context context, java.lang.String errorCode)	Произошла ошибка при взаимодействии с сервером. Параметры: - context – контекст; - errorCode – код ошибки.	Метод вызывается при возникновении ошибки взаимодействия с сервером.
6.	protected abstract void onReceive (Context context, Intent intent)	Получение нового push-сообщения из других источников. Пример: public void onReceive(Context context, Intent intent) { String from = intent.getStringExtra("key.from"); Bundle data = intent.getBundleExtra("key.new.push.parameters");}	Метод вызывается при получении нового push-сообщения из других источников. В качестве параметра передается текст, контекст, senderId сервера отправителя и bundle с информацией из push-уведомления

4.1.3.2. Обработчик сохранения сообщений PushServerIntentService

Для получения списка сообщений с push-сервера в наследнике класса PushServerIntentService требуется реализовать методы, представленные в Таблица 2.

Таблица 2. Методы класса PushServerIntentService.

№ п/п	Метод	Описание	Условия вызова
1.	protected abstract boolean saveMessages(final List<PushMessage> pushMessages);	Получен список сообщений с push-сервера. Параметры: - pushMessages – список сообщений. Возвращаемое значение true при успешном сохранении сообщений, иначе false	Метод вызывается при получении списка новых PushMessages с push-сервера
2.	protected abstract void messagesWereRead(final List<String> messageIds);	Получен список идентификаторов прочитанных push-сообщений. Параметры: - messageIds – список идентификаторов прочитанных push-сообщений	Метод вызывается при получении списка идентификаторов прочитанных push-сообщений

Примечание! Метод saveMessages вызывается не в main-thread.

4.1.4. Методы библиотеки

Класс PushController реализует методы для предоставления доступа к функциональности библиотеки. Описание методов представлено в Таблица 3.

Таблица 3. Методы класса PushController.

№ п/п	Метод	Описание
1.	register()	Данный метод инициализирует процедуру регистрации приложения на push-сервере
2.	getDeviceUid()	Данный метод инициализирует процедуру получения deviceUid от push-сервера
3.	setDeviceUid(java.lang.String deviceUid)	Данный метод используется при формировании deviceUid в приложении
4.	setPhoneNumber(java.lang.String phoneNumber)	Данный метод позволяет выполнить регистрацию приложения по phoneNumber синхронно
5.	setPhoneNumberAsync(java.lang.String phoneNumber, com.pushserver.android.RequestCallback<com.pushserver.android.SetPhoneNumberResult,com.pushserver.android.exception.PushServer ErrorException> callback)	Данный метод позволяет выполнить регистрацию приложения по phoneNumber асинхронно. Для регистрации приложения по phoneNumber также создан метод Push UI, который вызывает в приложении диалоговое окно для ввода номера телефона
6.	confirmPhoneNumber(java.lang.String confirmationCode)	Данный метод позволяет получить подтверждение значения phoneNumber синхронно
7.	confirmPhoneNumberAsync(java.lang.String confirmationCode, com.pushserver.android.RequestCallback<java.lang.Void,com.pushserver.android.exception.PushServerErrorException> callback)	Данный метод позволяет получить подтверждение значения phoneNumber асинхронно
8.	setClientId(java.lang.String clientId)	Данный метод позволяет выполнить регистрацию приложения по clientId синхронно
9.	setClientIdAsync(java.lang.String clientId, com.pushserver.android.RequestCallback<java.lang.Void,com.pushserver.android.exception.PushServerErrorException> callback)	Данный метод позволяет выполнить регистрацию приложения по clientId асинхронно
10.	getMessageHistory(int maxCount)	Получение истории сообщений. Метод загружает maxCount последних сообщений синхронно.

№ п/п	Метод	Описание
		Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
11.	getMessageHistoryAsync(int maxCount)	Получение истории сообщений. Метод загружает maxCount последних сообщений асинхронно. Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
12.	getNextMessageHistory(int maxCount	Получение истории сообщений. Метод загружает maxCount следующих сообщений (при первом вызове соответствует getMessageHistory) синхронно. Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
13.	getNextMessageHistoryAsync(int maxCount)	Получение истории сообщений. Метод загружает maxCount следующих сообщений (при первом вызове соответствует getMessageHistory) асинхронно. Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
14.	sendMessage(String context, boolean isSystem)	Синхронная отправка сообщения на push-сервер. isSystem = true, если сообщение системное (например typing). Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
15.	sendMessageAsync(String context, boolean isSystem)	Асинхронная отправка сообщения на push-сервер. isSystem = true, если сообщение системное (например typing). Для работы данного метода требуется обязательная привязка clientId к зарегистрированному приложению (история сообщений хранится с привязкой к clientId)
16.	sendFileAsync(final File file, final String password, final long validityTime, final RequestProgressCallback callback)	Отправка файла из приложения на сервер асинхронно

№ п/п	Метод	Описание
17.	resetCounterSync()	Данный метод позволяет обнулить значение счётчика новых push-сообщений на push-сервере
18.	notifyMessageUpdateNeeded()	Метод инициирует принудительную загрузку списка push-сообщений с push-сервера без предварительного получения push-уведомления
19.	notifyMessageUpdateNeeded(String sessionKey)	Метод для принудительной загрузки сообщений с push-сервера/ При наличии новых сообщений будет вызван PushServerIntentServer.saveMessages()
20.	notifyMessageRead(final String messageId)	Данный метод используется для уведомления push-сервера о прочтении push-сообщения
21.	setSubscriptionEnabled(boolean enabled)	Данный метод отправляет запрос на включение/отключение push-уведомлений в приложении
22.	isSubscriptionEnabled()	Данный метод возвращает текущее значение статуса подключения push-уведомлений в приложение
23.	getDeviceAddress()	Данный метод позволяет получить текущее значение deviceAddress на push-сервере
24.	getCurrentPushAddress()	Данный метод позволяет получить текущее значение deviceAddress в GCM
25.	subscribeDeviceAddressUpdates(PushController.TokenListener tokenListener)	Данный метод позволяет осуществить подписку приложения на события обновления его deviceAddress
26.	unSubscribeDeviceAddressUpdates(PushController.TokenListener tokenListener)	Данный метод позволяет осуществить отмену подписки приложения на события обновления его deviceAddress
27.	setCustomHeaderProcessor(HeaderProcessor headerProcessor)	Метод позволяет добавить дополнительные заголовки к сетевым запросам
28.	getVersion()	Метод возвращает текущую версию библиотеки

4.2. Регистрация приложения по deviceUid

Для регистрации приложения на push-сервере по deviceUid необходимо:

1. Вызвать метод PushController.getInstance(this).init() для инициализации библиотеки (например, в onCreate приложения).
2. Вызвать метод setDeviceUID(String deviceUID) для инициализации процедуру получения deviceUid.

Примечание! В случае если мета "com.pushserver.android.useDefaultDeviceUID" установлена в значении false, шаг 2 можно пропустить.

Библиотека выполнит вызов callback onDeviceAddressChanged, после вызова доступна процедура отправки push-сообщений по deviceUid.

4.3. Привязка clientId к зарегистрированному приложению

Для привязки clientId к зарегистрированному приложению необходимо:

1. Вызвать метод PushController.getInstance(this).init() для инициализации библиотеки (например, в onCreate приложения).
2. Вызвать метод PushController.getInstance(this).setClientIdAsync(String clientId, final RequestCallback<Void, PushServerErrorException> callback) для привязки clientId к приложению.

Библиотека выполнит вызов callback onDeviceAddressChanged, после вызова доступна процедура отправки push-сообщений по clientId.

Примечание! Для синхронной установки clientId сипользуется метод public void setClientId(final @Nullable String clientId) throws PushServerErrorException.

4.4. Привязка phoneNumber к зарегистрированному приложению

Для привязки phoneNumber к зарегистрированному приложению необходимо вызвать последовательно методы:

1. setPhoneNumberAsync – метод регистрации приложения по номеру телефона.
2. confirmPhoneNumberAsyn – метод подтверждения номера телефона.

5. СТРУКТУРА СООБЩЕНИЯ PUSHMESSAGE

Структура сообщения PushMessage представлена в Таблица 4.

Таблица 4. Структура сообщения PushMessage.

Имя	**Тип**	**Значение**
messageId	String	Идентификатор сообщения
sentAt	String	Дата и время формирования сообщения в формате dd.MM.yyyy HH:mm:ss Z, где Z – часовой пояс в формате RFC 822
shortMessage	String	Сокращенная форма сообщения, отправляется через push-сеть в виде системного push-уведомления (Alert). Предназначено для отображения в списке сообщений в виде заголовка
fullMessage	String	Полный текст push-сообщения (base64-encoded HTML)
secured	Boolean	Признак безопасного сообщения
read	Boolean	Статус прочтения сообщения
sessionKey	String	Сессионный ключ

Пример сообщения:

```
{messageId: "10",
sentAt: "24.10.2012 12:41:57",
shortMessage: "Уважаемый, ФИО! Изменились …",
fullMessage: "0KPQstCw0LbQsNC10LzRi9C5INC60LvQuNC10L3RgiEg0JjQt",
secured: true
read: true
sessionKey: sessionKey
}
```

6. КОДЫ ОШИБОК

В метод, обрабатывающий ошибки взаимодействия с push-сервером, передаются коды ошибок. Описание кодов ошибок представлено в Таблица 5.

Таблица 5. Список кодов ошибок.

№ п/п	Код ошибки	Описание
1.	DEVICE_ADDRESS_INVALID	Указано некорректное значение идентификатора deviceAddress
2.	ACCESS_DENIED	Недостаточно прав для выполнения операции
3.	INTERNAL_SERVER_ERROR	Внутренняя ошибка сервера: требуется повторить запрос позже
4.	BAD_PARAMETERS	Указаны неверные параметры запроса. Данная ошибка возникает в случае неверного значения deviceAddress или его отсутствия на push-сервере
5.	IO_ERROR	Сетевая ошибка
6.	CLIENT_ID_INVALID	Указано некорректное значение идентификатора clientId
7.	PHONE_NUMBER_INVALID	Указано некорректное значение идентификатора phoneNumber
8.	CONFIRMATION_CODE_INVALID	Указано неверное значение confirmationCode
9.	CONFIRMATION_CODE_EXPIRED	Истек срок действия confirmationCode
10.	PHONE_NUMBER_IS_EMPTY	Не указано значение номера телефона
11.	CONFIRMATION_CODE_IS_EMPTY	Не указано значение confirmation code
12.	JSON_FORMING_ERROR	Неверно сформировано json-сообщение

Примечание! При возникновение ошибок сетевого взаимодействия или некорректной передаче параметров методам errorCode > 0. Если ошибок нет, тогда errorCode = nil, либо errorCode = 0: errorCode:Int32 = 0.

Описание ошибки: errorDescription.